

EXHIBIT N

Oracle8*i*

Data Warehousing Guide

Release 2 (8.1.6)

December 1999

Part No. A76994-01

ORACLE

Data Warehousing Guide, Release 2 (8.1.6)

Part No. A76994-01

Copyright © 1996, 1999, Oracle Corporation. All rights reserved.

Primary Author: Paul Lane

Contributing Author: George Lumpkin

Contributors: Patrick Amor, Tolga Bozkaya, Karl Dias, Yu Gong, Ira Greenberg, Helen Grembowicz, John Haydu, Meg Hennington, Lilian Hobbs, Hakan Jakobsson, Jack Raitto, Ray Roccaforte, Andy Witkowski, Zia Ziauddin

Graphic Designer: Valarie Moore

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and Oracle disclaims liability for any damages caused by such use of the Programs.

The Programs (which include both the software and documentation) contain proprietary information of Oracle Corporation; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. Oracle Corporation does not warrant that this document is error free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Oracle Corporation.

If the Programs are delivered to the U.S. Government or anyone licensing or using the Programs on behalf of the U.S. Government, the following notice is applicable:

Restricted Rights Notice Programs delivered subject to the DOD FAR Supplement are "commercial computer software" and use, duplication, and disclosure of the Programs including documentation, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, Programs delivered subject to the Federal Acquisition Regulations are "restricted computer software" and use, duplication, and disclosure of the Programs shall be subject to the restrictions in FAR 52.227-19, Commercial Computer Software - Restricted Rights (June, 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

Oracle is a registered trademark, and Enterprise Manager, Pro*COBOL, Server Manager, SQL*Forms, SQL*Net, and SQL*Plus, Net8, Oracle Call Interface, Oracle7, Oracle7 Server, Oracle8, Oracle8 Server, Oracle8i, Oracle Forms, PL/SQL, Pro*C, Pro*C/C++, and Trusted Oracle are registered trademarks or trademarks of Oracle Corporation. All other company or product names mentioned are used for identification purposes only and may be trademarks of their respective owners.

Send Us Your Comments	xv
Preface.....	xvii
Audience.....	xvii
Knowledge Assumed of the Reader	xvii
Installation and Migration Information	xvii
Application Design Information	xvii
How Oracle8i Data Warehousing Guide Is Organized	xviii
Conventions Used in This Manual	xx

Part I Concepts

1 Data Warehousing Concepts

What is a Data Warehouse?	1-2
Subject Oriented.....	1-2
Integrated.....	1-2
Nonvolatile	1-2
Time Variant.....	1-3
Contrasting a Data Warehouse with an OLTP System	1-3
Typical Data Warehouse Architectures.....	1-5

Part II Logical Design

2 Overview of Logical Design

Logical vs. Physical.....	2-2
Create a Logical Design	2-2
Data Warehousing Schemas	2-3
Star Schemas.....	2-3
Other Schemas	2-4
Data Warehousing Objects.....	2-4
Fact Tables	2-5
Dimensions	2-5

Part III Physical Design

3 Overview of Physical Design

Moving from Logical to Physical Design.....	3-2
Physical Design	3-3
Physical Design Structures.....	3-3
Tablespaces.....	3-3
Partitions	3-3
Indexes.....	3-4
Constraints.....	3-4

4 Hardware and I/O

Striping	4-2
Input/Output Considerations	4-9
Staging File Systems	4-9

5 Parallelism and Partitioning

Overview of Parallel Execution Tuning.....	5-2
When to Implement Parallel Execution.....	5-2
Tuning Physical Database Layouts.....	5-3
Types of Parallelism	5-3
Partitioning Data.....	5-4
Partition Pruning	5-10
Partition-wise Joins.....	5-12

6 Indexes

Bitmap Indexes	6-2
B-tree Indexes	6-6
Local Versus Global.....	6-6

7 Constraints

Why Constraints are Useful in a Data Warehouse	7-2
Overview of Constraint States.....	7-2
Typical Data Warehouse Constraints	7-3
Unique Constraints in a Data Warehouse.....	7-3

Foreign Key Constraints in a Data Warehouse	7-5
RELY Constraints	7-5
Constraints and Parallelism	7-6
Constraints and Partitioning	7-6

8 Materialized Views

Overview of Data Warehousing with Materialized Views	8-2
Materialized Views for Data Warehouses	8-3
Materialized Views for Distributed Computing	8-3
Materialized Views for Mobile Computing	8-3
The Need for Materialized Views	8-3
Components of Summary Management	8-5
Terminology	8-7
Schema Design Guidelines for Materialized Views	8-8
Types of Materialized Views	8-10
Materialized Views with Joins and Aggregates	8-11
Single-Table Aggregate Materialized Views	8-12
Materialized Views Containing Only Joins	8-13
Creating a Materialized View	8-16
Naming	8-17
Storage Characteristics	8-17
Build Methods	8-18
Used for Query Rewrite	8-18
Query Rewrite Restrictions	8-18
Refresh Options	8-19
ORDER BY	8-22
Using Oracle Enterprise Manager	8-23
Nested Materialized Views	8-23
Why Use Nested Materialized Views?	8-23
Rules for Using Nested Materialized Views	8-24
Restrictions when Using Nested Materialized Views	8-24
Limitations of Nested Materialized Views	8-25
Example of a Nested Materialized View	8-26
Nesting Materialized Views with Joins and Aggregates	8-28
Nested Materialized View Usage Guidelines	8-28

Registration of an Existing Materialized View	8-29
Partitioning a Materialized View.....	8-31
Partitioning the Materialized View.....	8-32
Partitioning a Prebuilt Table	8-33
Indexing Selection for Materialized Views	8-34
Invalidating a Materialized View	8-34
Security Issues	8-35
Guidelines for Using Materialized Views in a Data Warehouse.....	8-35
Altering a Materialized View	8-36
Dropping a Materialized View.....	8-36
Overview of Materialized View Management Tasks.....	8-37

9 Dimensions

What is a Dimension?	9-2
Drilling Across	9-5
Creating a Dimension	9-6
Multiple Hierarchies.....	9-8
Using Normalized Dimension Tables.....	9-10
Dimension Wizard.....	9-11
Viewing Dimensions.....	9-11
Using The DEMO_DIM Package.....	9-12
Using Oracle Enterprise Manager	9-13
Dimensions and Constraints	9-13
Validating a Dimension	9-14
Altering a Dimension.....	9-14
Deleting a Dimension	9-15

Part IV Managing the Warehouse Environment

10 ETT Overview

ETT Overview.....	10-2
ETT Tools	10-2
ETT Sample Schema.....	10-3

11 Extraction

Overview of Extraction	11-2
Extracting Via Data Files	11-2
Extracting into Flat Files Using SQL*Plus.....	11-3
Extracting into Flat Files Using OCI or Pro*C Programs.....	11-4
Exporting into Oracle Export Files Using Oracle's EXP Utility	11-4
Copying to Another Oracle Database Using Transportable Tablespaces	11-5
Extracting Via Distributed Operations	11-5
Change Capture.....	11-6
Timestamps	11-7
Partitioning	11-7
Triggers	11-7

12 Transportation

Transportation Overview	12-2
Transportation of Flat Files	12-2
Transportation Via Distributed Operations.....	12-2
Transportable Tablespaces	12-3

13 Transformation

Techniques for Data Transformation Inside the Database	13-2
Transformation Flow.....	13-2
Transformations Provided by SQL*Loader	13-3
Transformations Using SQL and PL/SQL	13-4
Data Substitution	13-5
Key Lookups.....	13-5
Pivoting	13-7
Emphasis on Transformation Techniques	13-9

14 Loading and Refreshing

Refreshing a Data Warehouse	14-2
Using Partitioning to Improve Data Warehouse Refresh.....	14-2
Populating Databases Using Parallel Load.....	14-10
Refreshing Materialized Views	14-16

Complete Refresh.....	14-18
Fast Refresh.....	14-18
Tips for Refreshing Using Refresh	14-22
Complex Materialized Views.....	14-27
Recommended Initialization Parameters for Parallelism	14-27
Monitoring a Refresh.....	14-28
Tips after Refreshing Materialized Views.....	14-28
 15 Summary Advisor	
Summary Advisor	15-2
Collecting Structural Statistics	15-3
Collection of Dynamic Workload Statistics	15-3
Recommending Materialized Views.....	15-5
Estimating Materialized View Size	15-7
Summary Advisor Wizard	15-8
Is a Materialized View Being Used?.....	15-8
 Part V Warehouse Performance	
 16 Schemas	
Schemas.....	16-2
Star Schemas	16-2
Optimizing Star Queries	16-4
Tuning Star Queries.....	16-4
Star Transformation.....	16-5
 17 SQL for Analysis	
Overview.....	17-2
Analyzing Across Multiple Dimensions	17-2
Optimized Performance.....	17-4
A Scenario	17-5
ROLLUP	17-6
Syntax	17-6
Details.....	17-6

Example.....	17-6
Interpreting NULLs in Results	17-8
Partial Rollup	17-8
Calculating Subtotals without ROLLUP	17-9
When to Use ROLLUP	17-10
CUBE	17-10
Syntax	17-11
Details.....	17-11
Example.....	17-11
Partial Cube	17-13
Calculating Subtotals without CUBE	17-14
When to Use CUBE	17-14
Using Other Aggregate Functions with ROLLUP and CUBE.....	17-15
GROUPING Function.....	17-15
Syntax	17-15
Examples.....	17-16
When to Use GROUPING	17-18
Other Considerations when Using ROLLUP and CUBE.....	17-19
Hierarchy Handling in ROLLUP and CUBE.....	17-19
Column Capacity in ROLLUP and CUBE.....	17-20
HAVING Clause Used with ROLLUP and CUBE.....	17-20
ORDER BY Clause Used with ROLLUP and CUBE.....	17-21
Analytic Functions.....	17-21
Ranking Functions.....	17-24
Windowing Functions.....	17-35
Reporting Functions	17-43
Lag/Lead Functions.....	17-46
Statistics Functions	17-46
Case Expressions.....	17-52
CASE Example	17-52
Creating Histograms with User-defined Buckets	17-53
 18 Tuning Parallel Execution	
Introduction to Parallel Execution Tuning.....	18-2
When to Implement Parallel Execution.....	18-2

Initializing and Tuning Parameters for Parallel Execution	18-3
Selecting Automated or Manual Tuning of Parallel Execution	18-3
Automatically Derived Parameter Settings under Fully Automated Parallel Execution	18-4
Setting the Degree of Parallelism and Enabling Adaptive Multi-User	18-5
Degree of Parallelism and Adaptive Multi-User and How They Interact	18-5
Enabling Parallelism for Tables and Queries.....	18-6
Forcing Parallel Execution for a Session.....	18-7
Controlling Performance with PARALLEL_THREADS_PER_CPU	18-7
Tuning General Parameters.....	18-8
Parameters Establishing Resource Limits for Parallel Operations	18-8
Parameters Affecting Resource Consumption	18-17
Parameters Related to I/O.....	18-25
Example Parameter Setting Scenarios for Parallel Execution	18-27
Example One: Small Datamart.....	18-28
Example Two: Medium-sized Data Warehouse.....	18-29
Example Three: Large Data Warehouse	18-30
Example Four: Very Large Data Warehouse	18-31
Miscellaneous Tuning Tips.....	18-33
Formula for Memory, Users, and Parallel Execution Server Processes.....	18-33
Setting Buffer Pool Size for Parallel Operations.....	18-36
Balancing the Formula	18-37
Examples: Balancing Memory, Users, and Parallel Execution Servers.....	18-40
Parallel Execution Space Management Issues	18-43
Tuning Parallel Execution on Oracle Parallel Server.....	18-44
Overriding the Default Degree of Parallelism.....	18-48
Rewriting SQL Statements.....	18-49
Creating and Populating Tables in Parallel	18-50
Creating Temporary Tablespace for Parallel Sort and Hash Join	18-51
Executing Parallel SQL Statements	18-53
Using EXPLAIN PLAN to Show Parallel Operations Plans.....	18-54
Additional Considerations for Parallel DML	18-54
Creating Indexes in Parallel	18-57
Parallel DML Tips.....	18-59
Incremental Data Loading in Parallel.....	18-62
Using Hints with Cost-Based Optimization	18-64

Monitoring and Diagnosing Parallel Execution Performance	18-64
Is There Regression?.....	18-66
Is There a Plan Change?.....	18-67
Is There a Parallel Plan?.....	18-67
Is There a Serial Plan?	18-67
Is There Parallel Execution?	18-68
Is The Workload Evenly Distributed?	18-68
Monitoring Parallel Execution Performance with Dynamic Performance Views	18-69
Monitoring Session Statistics	18-72
Monitoring Operating System Statistics.....	18-75
 19 Query Rewrite	
Overview of Query Rewrite	19-2
Cost-Based Rewrite	19-3
Enabling Query Rewrite	19-4
Initialization Parameters for Query Rewrite	19-5
Privileges for Enabling Query Rewrite	19-6
When Does Oracle Rewrite a Query?	19-6
Query Rewrite Methods	19-8
SQL Text Match Rewrite Methods	19-8
General Query Rewrite Methods	19-9
Query Rewrite with CUBE/ROLLUP Operator	19-20
When are Constraints and Dimensions Needed?	19-21
Complex Materialized Views.....	19-21
View-based Materialized View	19-22
Rewrite with Nested Materialized Views	19-22
Expression Matching	19-23
Date Folding.....	19-24
Accuracy of Query Rewrite	19-26
Did Query Rewrite Occur?	19-27
Explain Plan.....	19-27
Controlling Query Rewrite	19-28
Guidelines for Using Query Rewrite	19-29
Constraints.....	19-29
Dimensions	19-29

Outer Joins 19-29

SQL Text Match..... 19-30

Aggregates 19-30

Grouping Conditions 19-30

Expression Matching..... 19-31

Date Folding 19-31

Statistics..... 19-31

Part VI Miscellaneous

20 Data Marts

What Is a Data Mart? 20-2

How Is It Different from a Data Warehouse?..... 20-2

Dependent, Independent, and Hybrid Data Marts 20-2

Extraction, Transformation, and Transportation 20-5

A Glossary

2

Overview of Logical Design

This chapter tells how to design a data warehousing environment, and includes the following topics:

- Logical vs. Physical
- Create a Logical Design
- Data Warehousing Schemas

Logical vs. Physical

If you are reading this guide, it is likely that your organization has already decided to build a data warehouse. Moreover, it is likely that the business requirements are already defined, the scope of your application has been agreed upon, and you have a conceptual design. So now you need to translate your requirements into a system deliverable. In this step, you create the logical and physical design for the data warehouse and, in the process, define the specific data content, relationships within and between groups of data, the system environment supporting your data warehouse, the data transformations required, and the frequency with which data is refreshed.

The logical design is more conceptual and abstract than the physical design. In the *logical design*, you look at the logical relationships among the objects. In the *physical design*, you look at the most effective way of storing and retrieving the objects.

Your design should be oriented toward the needs of the end users. End users typically want to perform analysis and look at aggregated data, rather than at individual transactions. Your design is driven primarily by end-user utility, but the end users may not know what they need until they see it. A well-planned design allows for growth and changes as the needs of users change and evolve.

By beginning with the logical design, you focus on the information requirements without getting bogged down immediately with implementation detail.

Create a Logical Design

A logical design is a conceptual, abstract design. You do not deal with the physical implementation details yet; you deal only with defining the types of information that you need.

The process of logical design involves arranging data into a series of logical relationships called entities and attributes. An *entity* represents a chunk of information. In relational databases, an entity often maps to a table. An *attribute* is a component of an entity and helps define the uniqueness of the entity. In relational databases, an attribute maps to a column.

You can create the logical design using a pen and paper, or you can use a design tool such as Oracle Warehouse Builder or Oracle Designer.

While entity-relationship diagramming has traditionally been associated with highly normalized models such as online transaction processing (OLTP) applications, the technique is still useful in dimensional modeling. You just approach it differently. In dimensional modeling, instead of seeking to discover

atomic units of information and all of the relationships between them, you try to identify which information belongs to a central fact table(s) and which information belongs to its associated dimension tables.

One output of the logical design is a set of entities and attributes corresponding to fact tables and dimension tables. Another output of mapping is operational data from your source into subject-oriented information in your target data warehouse schema. You identify business subjects or fields of data, define relationships between business subjects, and name the attributes for each subject.

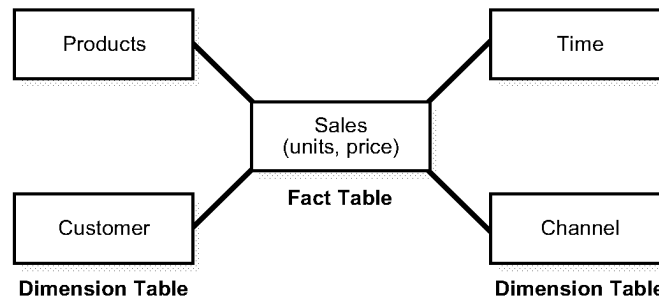
The elements that help you to determine the data warehouse schema are the model of your source data and your user requirements. Sometimes, you can get the source model from your company's enterprise data model and reverse-engineer the logical data model for the data warehouse from this. The physical implementation of the logical data warehouse model may require some changes due to your system parameters—size of machine, number of users, storage capacity, type of network, and software.

Data Warehousing Schemas

A schema is a collection of database objects, including tables, views, indexes, and synonyms. There are a variety of ways of arranging schema objects in the schema models designed for data warehousing. Most data warehouses use a dimensional model.

Star Schemas

The star schema is the simplest data warehouse schema. It is called a star schema because the diagram of a star schema resembles a star, with points radiating from a center. The center of the star consists of one or more fact tables and the points of the star are the dimension tables shown in Figure 2-1:

Figure 2–1 Star Schema

Unlike other database structures, in a star schema, the dimensions are denormalized. That is, the dimension tables have redundancy which eliminates the need for multiple joins on dimension tables. In a star schema, only one join is needed to establish the relationship between the fact table and any one of the dimension tables.

The main advantage to a star schema is optimized performance. A star schema keeps queries simple and provides fast response time because all the information about each level is stored in one row. See Chapter 16, "Schemas", for further information regarding schemas.

Note: Oracle recommends you choose a star schema unless you have a clear reason not to.

Other Schemas

Some schemas use third normal form rather than star schemas or the dimensional model.

Data Warehousing Objects

The following types of objects are commonly used in data warehouses:

- Fact tables are the central tables in your warehouse schema. Fact tables typically contain facts and foreign keys to the dimension tables. Fact tables represent data usually numeric and additive that can be analyzed and examined. Examples include Sales, Cost, and Profit.

- Dimension tables, also known as lookup or reference tables, contain the relatively static data in the warehouse. Examples are stores or products.

Fact Tables

A fact table is a table in a star schema that contains facts. A fact table typically has two types of columns: those that contain facts, and those that are foreign keys to dimension tables. A fact table might contain either detail-level facts or facts that have been aggregated. Fact tables that contain aggregated facts are often called **summary tables**. A fact table usually contains facts with the same level of aggregation.

Values for facts or measures are usually not known in advance; they are observed and stored.

Fact tables are the basis for the data queried by OLAP tools.

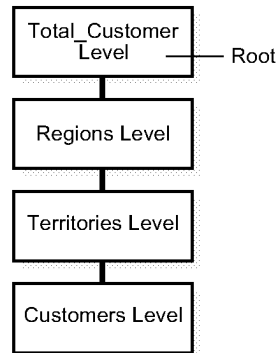
Creating a New Fact Table

You must define a fact table for each star schema. A fact table typically has two types of columns: those that contain facts, and those that are foreign keys to dimension tables. From a modeling standpoint, the primary key of the fact table is usually a composite key that is made up of all of its foreign keys; in the physical data warehouse, the data warehouse administrator may or may not choose to create this primary key explicitly.

Facts support mathematical calculations used to report on and analyze the business. Some numeric data are dimensions in disguise, even if they seem to be facts. If you are not interested in a summarization of a particular item, the item may actually be a dimension. Database size and overall performance improve if you categorize borderline fields as dimensions.

Dimensions

A dimension is a structure, often composed of one or more hierarchies, that categorizes data. Several distinct dimensions, combined with measures, enable you to answer business questions. Commonly used dimensions are Customer, Product, and Time. Figure 2-2 shows some a typical dimension hierarchy.

Figure 2–2 Typical Levels in a Dimension Hierarchy

Dimension data is typically collected at the lowest level of detail and then aggregated into higher level totals, which is more useful for analysis. For example, in the Total_Customer dimension, there are four levels: Total_Customer, Regions, Territories, and Customers. Data collected at the Customers level is aggregated to the Territories level. For the Regions dimension, data collected for several regions such as Western Europe or Eastern Europe might be aggregated as a fact in the fact table into totals for a larger area such as Europe.

See Chapter 9, "Dimensions", for further information regarding dimensions.

Hierarchies

Hierarchies are logical structures that use ordered levels as a means of organizing data. A hierarchy can be used to define data aggregation. For example, in a Time dimension, a hierarchy might be used to aggregate data from the Month level to the Quarter level to the Year level. A hierarchy can also be used to define a navigational drill path and establish a family structure.

Within a hierarchy, each level is logically connected to the levels above and below it; data values at lower levels aggregate into the data values at higher levels. For example, in the Product dimension, there might be two hierarchies—one for product identification and one for product responsibility.

Dimension hierarchies also group levels from very general to very granular. Hierarchies are utilized by query tools, allowing you to drill down into your data to view different levels of granularity—one of the key benefits of a data warehouse.

When designing your hierarchies, you must consider the relationships defined in your source data. For example, a hierarchy design must honor the foreign key relationships between the source tables in order to properly aggregate data.

Hierarchies imposes a family structure on dimension values. For a particular level value, a value at the next higher level is its parent, and values at the next lower level are its children. These familial relationships allow analysts to access data quickly.

See Chapter 9, "Dimensions", for further information regarding hierarchies.

Levels Levels represent a position in a hierarchy. For example, a Time dimension might have a hierarchy that represents data at the Month, Quarter, and Year levels. Levels range from general to very specific, with the root level as the highest, or most general level. The levels in a dimension are organized into one or more hierarchies.

Level Relationships Level relationships specify top-to-bottom ordering of levels from most general (the root) to most specific information and define the parent-child relationship between the levels in a hierarchy.

You can define hierarchies where each level rolls up to the previous level in the dimension or you can define hierarchies that skip one or multiple levels.

Data Warehousing Schemas

Part III

Physical Design

This section deals with physical design in a data warehouse.

It contains the following chapters:

- Overview of Physical Design
- Hardware and I/O
- Parallelism and Partitioning
- Indexes
- Constraints
- Materialized Views
- Dimensions

3

Overview of Physical Design

This chapter describes physical design in a data warehousing environment, and includes the following:

- Moving from Logical to Physical Design
- Physical Design

Moving from Logical to Physical Design

In a sense, logical design is what you draw with a pencil before building your warehouse and physical design is when you create the database with SQL statements.

During the physical design process, you convert the data gathered during the logical design phase into a description of the physical database, including tables and constraints. Physical design decisions, such as the type of index or partitioning have a large impact on query performance. See Chapter 6, "Indexes" for further information regarding indexes. See Chapter 5, "Parallelism and Partitioning" for further information regarding partitioning.

Logical models use fully normalized entities. The entities are linked together using relationships. Attributes are used to describe the entities. The UID distinguishes between one instance of an entity and another.

A graphical way of looking at the differences between logical and physical designs is in Figure 3-1:

Figure 3-1 Logical Design Compared with Physical Design

Logical	Physical
Entity	Table
Relationship	Foreign Key
Attribute	Column
Unique Identifier	Primary Key

Physical Design

Physical design is where you translate the expected schemas into actual database structures. At this time, you have to map:

- Entities to Tables
- Relationships to Foreign Keys
- Attributes to Columns
- Primary Unique Identifiers to the Primary Key
- Unique Identifiers to Unique Keys

You will have to decide whether to use a one-to-one mapping as well.

Physical Design Structures

Translating your schemas into actual database structures requires creating the following:

- Tablespaces
- Partitions
- Indexes
- Constraints

Tablespaces

Tablespaces need to be separated by differences. For example, tables should be separated from their indexes and small tables should be separated from large tables. See Chapter 4, "Hardware and I/O", for further information regarding tablespaces.

Partitions

Partitioning large tables improves performance because each partitioned piece is more manageable. Typically, you partition based on transaction dates in a data warehouse. For example, each month. This month's worth of data can be assigned its own partition. See Chapter 5, "Parallelism and Partitioning", for further details.

Indexes

Data warehouses' indexes resemble OLTP indexes. An important point is that bitmap indexes are quite common. See Chapter 6, "Indexes", for further information.

Constraints

Constraints are somewhat different in data warehouses than in OLTP environments because data integrity is reasonably ensured due to the limited sources of data and because you can check the data integrity of large files for batch loads. Not null constraints are particularly common in data warehouses. See Chapter 7, "Constraints", for further details.